## REMARKS

Claims 1-16 are pending. Claim 2 was amended to correct a grammatical error. Claim 4 was amended to correct antecedent basis issues. Claim 6 was amended for further clarification. Claims 10-16 were added to further define the present invention.

### Support for new claim language

None of the amendments introduce new matter. Support for all of the new claim language exists in preferred embodiments disclosed in the present specification, as follows:

Claim 6: Section X.7.5 on pages 61-62

Claim 10: Page 15, lines 4-5

Claim 11: Page 19, lines 3-6

Claim 12: Page 19, lines 25-27 and section V on page 46

Claim 13: Page 22, lines 17-20

Claims 14-16: Page 20, line 14 through page 21, line 29

### Request for Interview Prior to Formal Action on Amendment

Applicants request an interview prior to formal action on this response. An "Applicant Initiated Interview Request Form" accompanies this response. Please contact Applicants' undersigned representative to schedule the interview.

### Prior Art Rejection

Claims 1, 2, 4 and 8 were rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by Son et al., hereafter, "Son."

Claims 3 and 5 were rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Son in view of Arevalo et al.

Claim 9 was rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Son in view of Holliday et al., hereafter, "Holliday."

Claims 6-7 were rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by Holliday.

Applicants respectfully traverse each of these rejections.

1. Son

Son describes a replication control process for use with data objects and token sites. As described on page 145, section 2.1 of Son, a "token site" is a site that has a "token copy" of a data object. When performing a write operation, the data object has two values, namely, an "after value" (the new version) and a "before value" (the old version). As described in section 2.3 of Son, "update transactions," which are "write operations," are performed by using a "coordinator" and a "two-phase protocol." The coordinator maintains a list of the write operations that are requested by each of the token sites. As described in page 146, column 1, lines 34-49 of Son, update transactions are performed using the two phase protocol as follows:

i. First phase (vote phase): An update transaction sends an update message to each token site of every data object in its write-set. The update transaction waits until it gets a response from all of the token sites for each data object. If all token sites vote "YES," then the update transaction enters the second phase. Son does not explain what a "YES" vote means, but it is presumed that a "YES" vote indicates that a token site is in a state where it can commit.

ii. Second phase (commit phase): The update transaction sends the "after value"[1] of each data object to be written to the respective token sites.

In Son, no update transactions are performed at any token site until after all of the token sites have indicated that they can commit and the "after values" are sent to each of the token sites. The coordinator tracks which update transactions need to be performed, but no actual update transactions are performed at any token site until the commit phase.

[1] Son's reference to an "actual value" on page 146, column 1, line 43 is a typographical error and clearly should have read "after value." Page 146, column 1, lines 6-23 of Son describe a "before value" and an "after value." There is no "actual value" described in Son, nor does an "actual value" make any sense in the context of the sentence on lines 41-44 of Son.

2. Patentability of claims 1 and 4 over Son

a. Order of transaction steps differs between Son and claimed invention

In the outstanding rejection, the Examiner appears to be equating elements in Son with certain claim limitations as follows:

| Son | Applicant's claims |
|---|---|
| token site | node |
| data object | database |
| update message | ready to commit token |

Applicants disagree that Son's elements can be equated with Applicant's claimed elements as set forth above. However, Applicants will respond to the rejection under the presumption that the analogy is proper.

In claim 1, transactions are performed at an originating node (step (a)) at least up to a commit operation (step (b)(i)) before a decision is made at the other nodes that their respective databases are prepared for a commit operation (step (b)(iv)). Likewise, in claim 4, transactions are performed at a first node (first clause of step (a)) at least up to a commit operation (second clause of step (a)) before a decision is made at a second node that the target database is prepared for a commit operation (step (d)). That is, the originating or first node in claims 1 and 4 does not delay performing transactions until it finds out if the other nodes or second node are prepared to commit.

In contrast to the scheme in claims 1 and 4, Son does not perform any update transactions at any token sites (a token site is a site that has a copy of a data object – page 145, column 1, lines 24-25) until after a decision is made that each token site is prepared for a commit operation. Son both performs and commits update transactions only after each token site confirms that it can commit. This is clearly explained in page 146, column 1, lines 41-44 of Son, which reads as follows (underlining added for emphasis):

> If all token sites vote YES, then the transaction enters the second phase (*commit phase*). It sends the actual[2] [after] value of each data object to be written to the respective token sites.

No transaction can be performed at a token site unless the token site knows what the new data value should be, here, referred to as the "after" value.

Accordingly, Son does not disclose or suggest performing transactions in the order set forth in claims 1 and 4, namely, before a decision is made that databases at other nodes are prepared for a commit operation.

Applicants' scheme is not merely an obvious reordering of steps. There are significant advantages to Applicants' scheme. One advantage is that in Applicants' scheme, a plurality of transactions can be performed at the originating or first node without needing to determine whether the other nodes or second node are prepared for a commit operation. In Son, it must first be determined whether each of the token sites are prepared for a commit operation before any transactions can be performed. Applicants' scheme is thus more efficient than Son's scheme. Another advantage is that Applicants' scheme avoids collisions, whereas Son's scheme does not avoid collisions (referred to as "conflicts" in Son). Son even explicitly states so on page 146, column 1, lines 49-51, which reads as follows (underlining added for emphasis):

> An update transaction that executes its commit phase can never be aborted, even if it potentially conflicts with another transaction.

b. Replication engine is at each node in claim 4

Claim 4 recites that each node includes a replication engine, and that the replication engine assigns the ready to commit token (step (b)), sends the ready to commit token (step (c)), and determines whether a node is prepared for a commit (step (d)). Son relies on a completely different scheme wherein a central coordinator directs a two-phase protocol for a plurality of token sites. The coordinator in Son creates and sends the update message. Nowhere does Son disclose or suggest that any of the token sites include a replication engine. Thus, Son has a

[2] As discussed above, Son's reference to an "actual value" is a typographical error and clearly should have read "after value."

completely different architecture for performing data replication than the claimed architecture. By eliminating the need for a coordinator, Applicants' scheme allows for data replication to occur with less transmission paths and with less messaging. This advantage is discussed throughout the present specification in contrasting Applicants' scheme to Oracle's two-phase commit which also uses a coordinator in a similar manner as Son.

c. Summary of patentable limitations

To summarize, Son does not disclose or suggest at least the following underlined limitations in claims 1 and 4:

Claim 1

(a) initiating and performing transactions[3] to be executed in a database at an originating node; and
(b) replicating the transactions to at least one or more other nodes by:
(i) pausing each transaction being executed in the database at the originating node prior to a commit operation for the transaction,
(ii) assigning a ready to commit token to the transaction

Claim 4

A method of replicating data associated with a plurality of transactions in a data replication system including a plurality of first and second nodes connected via communication media in a topology, each node including (i) a database, (ii) a replication engine which performs data replication functions, and (iii) an application which executes transactions and posts the transactions to the database, the application being independent of the replication engine, each transaction being one or more transaction steps or transaction operations, the method comprising:
(a) an application at a first node initiating and performing transactions to be executed in a database at the first node, the application pausing each transaction being executed in the database at the first node prior to a commit operation for the transaction;
(b) a replication engine at the first node assigning a ready to commit token to the transaction in coordination with the application; (b) a replication engine at the first node assigning a ready to commit token to the transaction in coordination with the application;

[3] To be clear, Applicants' argument in claims 1 and 4 is not that Son does not perform transactions, but that Son does not perform transactions until after it is determined that all token sites can commit, whereas in claims 1 and 4, transactions are performed before it is determined that other nodes or a second node can commit.

(c) the replication engine at the first node sending the ready to commit token to the second node;
(d) a replication engine at a second node determining whether a target database at the second node is prepared for a commit operation for the transaction corresponding to the ready to commit token, and, if so, sending back the ready to commit token to the first node

For at least the reasons set forth above, claims 1 and 4 are believed to be patentable over Son.

3. Patentability of claim 8 over Son

Claim 8 reads as follows (underlining added for emphasis):

8. A method of performing dual writes for replicating transactions among plural databases located at different nodes, each transaction being one or more transaction steps or transaction operations, at least some of the transaction steps or transaction operations being update steps or operations which are performed only after database locking occurs, the method comprising:
(a) initiating a transaction at an originating node; and
(b) the dual write replication process causing database locking to occur at one or more other nodes only upon the occurrence of an update step or operation in the transaction at the originating node.

Ideally, it is preferable to have access to all data elements in a database at all times. Placing locks on data elements detracts from this goal but is a necessary evil to avoid database synchronization problems, such as collisions.

Section X. 7.4-7.5 on pages 60-63 of the present specification describes one preferred embodiment of the present invention wherein read locks are propogated only if a data element is subsequently updated. Accordingly, there is no transmission or corresponding locks at the replicated nodes unless an update occurs. Since many read operations do not have a subsequent update operation, such as a write operation, this scheme reduces transmission overhead (i.e., there is a reduction in messages among nodes to perform a lock operation) and also reduces the number of read locks that have to be performed.

In the outstanding Office Action, the Examiner asserts that Son's two-phase protocol clearly indicates that Son's dual write replication process causes database locking to occur at one

or more other nodes only during an update step (e.g., commitment of an update transaction) at the originating node. Applicants respectfully disagree.

Database locking likely occurs during Son's two-phase protocol. However, Son's two-phase protocol is used only when there is a write request. Nowhere does Son discuss whether or not database locking occurs during other types of requests, such as read-only requests. Stated simply, the mere fact that database locking likely occurs in Son during write requests does not mean that database locking occurs only during write requests in Son. It is conventional to perform database locking during read-only requests when there could potentially be a subsequent update (write request). Applicants' claimed invention includes a limitation that during a dual write replication process, database locking occurs at one or more other nodes only upon the occurrence of an update step or operation in the transaction at the originating node. Thus, there would be no locking during the read-only request, even if there could potentially be a subsequent update (write request).

To formulate a *prima facie* rejection of claim 8, Son must disclose or suggest this claim limitation. A disclosure of how locking occurs during a write request is not a disclosure that locking does not occur during non-write requests, as presumed by the Examiner. In fact, Son extensively discusses read-only requests throughout the document, but never discusses whether or not locking occurs during such read-only requests. Again, since it is conventional to lock during read-only requests when there could potentially be a subsequent update (write request), nothing can be read into Son's silence regarding whether or not locking occurs during such read-only requests.

For at least the reasons set forth above, claim 8 is believed to be patentable over Son.

4. Patentability of claim 6 over Holliday

Claim 6 reads as follows (underlining added for emphasis):

> 6. A method of performing dual writes for replicating transactions among plural databases located at different nodes, each transaction being one or more transaction steps or transaction operations, at least some of the transaction steps or transaction operations being update steps or operations, the method comprising:
> (a) initiating a transaction at an originating node;

> (b) inhibiting the dual write replication process from communicating transaction steps or operations of the transaction with one or more other nodes until an update step or operation occurs within the transaction at the originating node; and
> (c) upon the occurrence of the update step or operation and regardless of whether a commit operation has occurred, performing the dual write replication process with respect to the one or more other nodes and sending with the update step or operation all transaction steps or operations for the transaction that have occurred prior to the update step or operation for the transaction, or prior to the previous update step or operation if a previous update step or operation existed for the transaction.

In the outstanding Office Action, the Examiner repeats the previous rejection of claim 6 over Protocol A3 (delayed broadcast writes) described on page 160, column 1 of Holliday. Applicants respectfully traverse this rejection. As explained in the previous response, Protocol A3 of Holliday does not perform dual write replication upon the occurrence of an update step or operation. In fact, Protocol A3 of Holliday teaches against this limitation because update operations are deferred until commit time, when a single message with all updates is sent to all other sites. Holliday further explains that "a write operation is deferred until $T_i$ is ready to commit." Thus, in Protocol A3 of Holliday, no dual write replication is performed upon the occurrence of an update step or operation. At best, Protocol A3 of Holliday discloses performing dual write replication upon the occurrence of a "commit."

In response to Applicants' arguments, the Examiner asserts that:

> ...when a write or update *command* is received at a database, no data is actually updated until the commit time of operation. Therefore, an update *step or operation* (e.g., the actual writing of data) does not occur until the commit time of the update command.

This statement is incorrect. In fact, a write/update command causes data to be applied (but not committed) to a database. Holliday is actually describing a non-conventional scheme where update operations (which cause data to be applied to a database) are deferred until commit time.

Notwithstanding this fact, and in the interest of advancing prosecution of the present application, claim 6 was amended to explicitly recite that upon the occurrence of the update step or operation, the dual write replication process is performed regardless of whether a commit

operation has occurred. In Holliday, the dual write replication process is not performed unless a commit operation has occurred.

For at least the reasons set forth above, claim 6 is believed to be patentable over Holliday.

5. Patentability of previously presented dependent claims

The dependent claims are believed to be allowable because they depend upon respective allowable independent claims, and because they recite additional patentable steps.

6. Patentability of new dependent claims 10-16

Claim 10: This claim is believed to be patentable over Son because Son relies upon a coordinator to send the "update message" to each token site, whereas in Applicants' scheme, the transaction originating node sends the ready to commit token to the one or more other nodes. Son's coordinator is not a transaction originating node. Son's coordinator merely receives instructions from somewhere else that some entity wishes to perform a transaction.

Claim 11: This claim is believed to be patentable over Son because Son requires a YES vote from all token sites.

Claim 12: This claim is believed to be patentable over Son because Son does not disclose or suggest that its update message can be sent to a token site with any transaction data.

Claim 13: This claim is believed to be patentable over Son because Son does not disclose or suggest that any of the token sites can be indirectly connected to other token sites.

Claims 14-16: These claims are believed to be patentable over Son because Son does not disclose or suggest anything equivalent to the claimed "ready to sync" token.

Furthermore, the new dependent claims are also believed to be allowable because they depend upon respective allowable independent claims, and because they recite additional patentable steps.

**Conclusion**

Insofar as the Examiner's rejections were fully addressed, the instant application is in condition for allowance. Issuance of a Notice of Allowability of all pending claims is therefore earnestly solicited.

Respectfully submitted,

BRUCE D. HOLENSTEIN et al.

August 7, 2006 By: Clark Jablon
(Date)

CLARK A. JABLON
Registration No. 35,039
AKIN GUMP STRAUSS HAUER & FELD LLP
One Commerce Square
2005 Market Street - Suite 2200
Philadelphia, PA 19103-7086
Telephone: (215) 965-1200
Direct Dial: (215) 965-1293
Facsimile: (215) 965-1210
E-Mail: cjablon@akingump.com

CAJ:gem